Data Analyst Nanodegree
December Cohort 2014
Fang
November 2015

Enron POI Detection: Free-Response Questions

Question 1: Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]

- The enron project primarily provides the classification of persons of interest (poi) where the challenge lies in the limited and spotty nature of the data. Machine learning is useful to discover hidden insights programmatically without excessive explicit guidance. There are only 18 identified poi, and 127 non-poi for a total of 145 data points plus the ‘TOTAL’ outlier which provided a summation of the other features (146 total data points in the original data). Since ‘TOTAL’ is not a real person, it was removed from the dataset before the classification began.

- For each of the 145 data points there were 21 features including ‘poi’ in the original dataset with most of them missing values (Table 1). The most sparse feature is ‘loan_advances’ with 142 missing values out of 145.

Table 1. Features and Missing Values (NaN)

Features	NaNs	Features	NaNs
‘bonus'	64	‘long_term_incentive'	80
‘deferral_payments'	1072	‘other'	53
‘deferred_income'	97	‘poi'	0
‘director_fees'	129	‘restricted_stock'	36
‘email_address'	35	‘restricted_stock_deferred'	128
‘exercised_stock_options'	44	‘salary'	51
‘expenses'	51	‘shared_receipt_with_poi'	60
‘from_messages'	60	‘to_messages'	60
‘from_poi_to_this_person'	60	‘total_payments'	21
‘from_this_person_to_poi'	60	‘total_stock_value'	20
‘loan_advances'	142		

Question 2: What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that doesn't come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

- Final Features:
  ['poi','exercised_stock_options','deferred_income','expenses']

- I performed Point Biserial Correlation using between the 'poi' label and each of the remaining 22 features (20 plus 3 created with 'email_address' removed). I selected the top 13 most correlated features with 'poi' which have a ~98% confidence level for those r-values. Beyond these top features the confidence interval drops and the significance of the correlation rapidly declines (see Table 2). The selected features are highlighted in yellow, and the correlation function implemented is *corrPOI()* and leverages *scipy.stats.pearsonr* which provides the same results as point biserial correlation but can be used for non-binary features.

- I split these 13 features into Financial and E-mail features and tested the two feature sets with a selection of classifiers. Financial features performed better than e-mail features in many of the classifiers tested, although this could be that there are more Financial features than E-mail features (see Table 2).

- In the lessons and after running classification tests, I found that e-mail features *'to_messages', 'from_messages', 'from_poi_to_this_person'* and *'from_this_person_to_poi'* were not a good features for differentiating between poi and non-poi. I created 3 new e-mail class features: *'to_poi_ratio'*, *'from_poi_ratio'* and *'shared_poi_ratio'*. They are the ratio of received or sent messages from poi and the total number of sent or received messages per person. These new features have much better correlation to *'poi'* and performed better when ran through the classifiers (see Tables 2 and 3). I originally used these new features in my final feature set consisting of *'exercised_stock_options', 'shared_receipt_with_poi', 'to_poi_ratio' and 'expenses'* but a set of financial features performed marginally better after an exhaustive search.

Table 2. Feature Correlations with ‘poi’ Label

Features	Correlation (r-value)	Significance (p-value)
poi	1	0
exercised_stock_options	0.3882409325	1.40E-06
total_stock_value	0.3841273288	1.84E-06
bonus	0.3602618794	8.55E-06
salary	0.3413652721	2.65E-05
to_poi_ratio	0.3248767325	6.70E-05
deferred_income	-0.275364467	0.0008017876245
long_term_incentive	0.2583005321	0.001707798738
shared_poi_ratio	0.2494845786	0.002476545285
restricted_stock	0.2493524798	0.00249013485
total_payments	0.242922426	0.003239087411
shared_receipt_with_poi	0.2421050642	0.003347588014
loan_advances	0.220405154	0.00772404163
expenses	0.2065801384	0.01266774391
from_poi_to_this_person	0.1915494782	0.0209984607
other	0.1701529037	0.04074414137
from_poi_ratio	0.1500505174	0.07163635871
from_this_person_to_poi	0.1303187174	0.11820945
to_messages	0.1100062418	0.1877759531
restricted_stock_deferred	-0.02122934524	0.7999195728
from_messages	-0.03330240285	0.6908855715
deferral_payments	-0.03826652615	0.6476931418
director_fees	-0.1200006845	0.1505209961

- The methodology I employed is that of a statistics aided exhaustive search. As a starting point and baseline, I loaded the entire feature set expanded with my created e-mail features totalling to 22 features and recorded the precision and recall. I then removed the uncorrelated features reducing the feature set to 13 (see Table 1). Each of the 13 features was then removed from the set starting with the lowest correlated and the precision and recall recorded as the evaluation metrics (Table 4).

- The delta change in precision and recall were then calculated for each feature removal and averaged across the classifiers where a negative delta indicated that the feature

removed had an adverse effect on the precision and recall score (see Appendix I for full selection process, separate file *'Appendix I.pdf'*).

- By using these delta averages, I determined the removal of 7 features that adversely affected the precision and recall. Using the same process, an exhaustive search was performed by removing features that when removed individually adversely affected the precision and recall of the classifiers (Appendix I). The classifiers and features that returned the highest precision and recall were noted and investigated further. The highest combined precision and recall score were found using the decision tree classifier with three financial features: '*exercised_stock_options', 'deferred_income', 'expenses'.*

Table 3. Comparison of Financial and E-mail Features (Original and Created)

	GuassianNB		Decision Tree		Random Forest		AdaboostDTC	
Features:	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Financial (10 Features)	0.18703	0.2235	0.27	0.275	0.41873	0.152	0.27778	0.28
E-mail (3 Features)	0.09625	0.1	0.2599	0.302	0.29922	0.231	0.23892	0.275
Original Email Features	0.01023	0.009	0.19103	0.277	0.18253	0.117	0.19343	0.271
Created Email Features	0.18796	0.206	0.23451	0.318	0.31638	0.224	0.23229	0.318

- No feature scaling was used in the final classification as feature scaling does not affect decision trees.

- The gini importance for the decision tree features are as follows:
  'exercised_stock_options': 0.332
  'expenses': 0.422
  'deferred_income': 0.245

Table 4: Determine Features that Adversely Affects Classifier Performance

	GuassianNB		Decision Tree		Random Forest		AdaboostDTC	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
All Features (22 Features)	0.1490	0.842	0.26385	0.25	0.42205	0.134	0.26854	0.248
Correlated (13 Features)	0.2474	0.3205	0.27283	0.245	0.46176	0.163	0.2525	0.227
Remove 'expenses'	0.2468	0.3205	0.23977	0.211	0.41514	0.148	0.23185	0.206
Remove 'loan_advances'	0.3735	0.31	0.23434	0.2095	0.45974	0.177	0.23568	0.214
'shared_receipt_with_poi'	0.3864	0.3105	0.22871	0.2095	0.42053	0.168	0.22767	0.204
Remove 'total_payments'	0.4360	0.349	0.26133	0.2335	0.41055	0.1595	0.24447	0.21
'restricted_stock'	0.4533	0.3495	0.27362	0.2505	0.48579	0.1795	0.27519	0.2485
'shared_poi_ratio'	0.46376	0.3615	0.27887	0.256	0.4386	0.159	0.28534	0.2695
'long_term_incentive'	0.4772	0.351	0.27797	0.2845	0.47984	0.1845	0.27189	0.281
'deferred_income'	0.45558	0.3	0.28491	0.3305	0.45622	0.211	0.2945	0.34
Remove 'to_poi_ratio'	0.5031	0.323	0.32525	0.3355	0.49748	0.247	0.33269	0.343
Remove 'salary'	0.4858	0.351	0.36441	0.383	0.58529	0.2985	0.36767	0.389
Remove 'bonus'	0.46889	0.2675	0.21564	0.193	0.32959	0.176	0.21928	0.199
'total_stock_value'	0.4605	0.321	0.27434	0.31	0.26912	0.278	0.27434	0.31

Question 3: What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: “pick an algorithm”]

- My final classifier was the decision tree classifier. I ran a number of untuned tests with a variety of feature sets against the naive bayes, decision tree, random forest, adaboost decision tree. Naive bayes and random forest classifiers generally had higher precision but lower recall with precision at times twice that of recall. Adaboost provided more even precision and recall scores that ran very close to pure decision tree scores. The tuning portion of the project made adaboost less desirable as the algorithm tended to take very long to execute. Decision trees seemed more stable in their performance and behaved more well with the addition and removal of features.

- I implemented *test_classifiers()* function that can be made to call a number of classifiers, primarily I used the four listed above, but the function also ran principal component analysis transformed decision tree. The function *featIter()* aided in performing the exhaustive search for the best classifier and features.

Question 4: What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well? How did you tune the parameters of your particular algorithm? (Some

algorithms don't have parameters that you need to tune--if this is the case for the one you picked, identify and briefly explain how you would have done it if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

- Many classifiers have parameters that can be adjusted to obtain better performance/results. Some algorithms have parameters that allows it to overcome local minimums or run many more iterations for an improved result. A algorithm that is not tuned well could be prone to overfitting or underfitting or even not produce a relevant result. Decision tree classifiers have several parameters that can be adjusted: *max_depth, min_samples_split, max_leaf_nodes, max_features*. If the *max_depth* parameter is chosen to be very low, the decision tree would terminate before all leaves will be pure refusing to continue splitting as needed.

- I tuned the decision tree classifier parameter *min_samples_split* manually with my own function *paramTune()* after being unable to gain much insight from various decision tree parameters tuned in the gridSearchCV algorithm implemented in sklearn (functions implemented *pcadtcGrid(), dtcGrid()*). I iterated over values of *min_samples_split* from 2 to 36 and found a maximum precision and recall at 3. Given the low *min_samples_split* I decided to take a look at *max_depth* for fear of overfitting. I tuned *max_depth* using my own function and found that around 10 to 12 *max_depth,* the precision and recall scores maxed and plateaued but otherwise did not improve the scores so the parameter was not specified in the final classifier.

Question 5: What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

- Validation is the testing of your model by splitting your available data into non-overlapping training and test sets. This is important because you want to be able to optimize the performance of a classifier with the available data before putting it to the test in the real world. A classic mistake would be to include some of the same data in the test and training sets thus artificially inflating the performance of the classifier. Another mistake for validation is to not perform cross validation where many test and training pairs are generated and scored. Because the enron data has very few data points and the labeling is unbalanced a modified version of the StratifiedShuffleSplit function found in tester.py was used to perform K-Fold cross validation.

- Precision and recall are used as metrics of evaluation because accuracy have certain shortcomings when the data is unbalanced. For datasets with skewed classes, in which vastly more examples of one class than the other exist, optimizing accuracy could cause high classifiers to only guess the more represented class in order to raise the performance metric resulting in really poor classification of the lesser represented class. Also in cases where the classifier needs to err on the side of more heavily classifying one class over the other accuracy may not be the best metric of evaluation.

- Precision and recall are measures of relevance. Precision is the accuracy of a classifying a specific element/class given that a particular element is determined: true positives/(true_positives + false positives). Recall is the accuracy of labeling a particular element given that it is that particular element: true_positive/(true_positives+false_negatives)

- In terms of the project and POI detection, Precision is the probability that given the classifier predicts a POI that it actually is a POI, or given that a POI is identified, the probability that it is correct. Recall in terms of POI detection is the probability that the classifier will correctly label a POI. Stated in reverse, 1 - Recall yields the probability that a POI is not identified correctly as POI.

- The final precision and accuracy for my Decision Tree Classifier with min_samples_split tuned to 3 using tester.py is highlighted in Table 5 along with the 2 other best classifiers and feature sets found.

Table 5. Three Best Classifiers and Feature Sets for POI detection

Classifier	DCT(min_samples_split=3)	PCA-DCT	DCT(min_samples_split=16)
Features	‘exercised_stock_options', 'deferred_income', 'expenses'	‘exercised_stock_options', 'to_poi_ratio', 'shared_receipt_with_poi'	‘exercised_stock_options', 'to_poi_ratio', 'shared_receipt_with_poi', 'expenses'
Precision	0.52973	0.54213	0.48328
Recall	0.49	0.46	0.4915
Accuracy	0.865	0.84525	0.85229

References

https://wiki.python.org/moin/UsingPickle

http://scikit-learn.org/stable/modules/pipeline.html

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

http://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html#example-ensemble-plot-adaboost-twoclass-py

https://en.wikipedia.org/wiki/Support_vector_machine

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

http://stackoverflow.com/questions/25475465/how-to-normalize-with-pca-and-scikit-learn

http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

http://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-analysis

http://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html#example-calibration-plot-compare-calibration-py

https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/122061/relationship-between-gini-importance-and-prediction-performance-say-auc

https://en.wikipedia.org/wiki/Precision_and_recall

https://en.wikipedia.org/wiki/Gini_coefficient

http://changingminds.org/explanations/research/analysis/choose_correlation.htm

http://stackoverflow.com/questions/3949226/calculating-pearson-correlation-and-significance-in-python

http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/

https://en.wikipedia.org/wiki/Null_hypothesis

http://blog.minitab.com/blog/adventures-in-statistics/how-to-correctly-interpret-p-values

http://www.cs.waikato.ac.nz/~mhall/thesis.pdf

https://en.wikipedia.org/wiki/Occam%27s_razor

http://scikit-learn.org/stable/modules/feature_selection.html

https://statswithcats.wordpress.com/2010/11/28/secrets-of-good-correlations/

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler

http://scikit-learn.org/stable/auto_examples/feature_selection/feature_selection_pipeline.html#example-feature-selection-feature-selection-pipeline-py

“I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.”